NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN APPROACH TO THE CONCEPT OF ERROR RECOVERY IN THE NATIONAL STOCK NUMBER SYSTEM

by

Mehmet YENEN

June 1986

Thesis Advisor: Harold Fredricksen

Approved for public release; distribution is unlimited.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)

6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b OFFICE SYMBOL *(If applicable)*	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000		7b ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION	8b OFFICE SYMBOL *(If applicable)*	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (*City, State, and ZIP Code*)		10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

11 TITLE (*Include Security Classification*)
AN APPROACH TO THE CONCEPT OF ERROR RECOVERY IN THE NATIONAL STOCK NUMBER SYSTEM

12 PERSONAL AUTHOR(S)
Mehmet Yenen

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (*Year, Month, Day*)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1986 June 20	63

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	

19 ABSTRACT (*Continue on reverse if necessary and identify by block number*)

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION
22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (*Include Area Code*) / 22c OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (If applicable) 52	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11 TITLE (Include Security Classification) Unclassified
An Approach to the Concept of Error Recovery in the National Stock Number System

12 PERSONAL AUTHOR(S)
Mehmet YENEN

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Masters Thesis	FROM _____ TO _____	1986 June 20	63

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Reed-Solomon Codes, Finite Field Theory, Algebraic Coding Theory, National Stock Number System, Systematic Error Correction

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis pertains to the area of systematic error recovery for the National Supply System. By techniques applied to real stock numbers a correction algorithm is successfully developed. The main research area for this thesis is that of algebraic coding theory, especially Reed-Solomon codes and their application for the National Supply System using finite field theory and the RS code over GS(11).
The findings of the research are also discussed for a possible database interface. This product of the stydy may be used to assist supply officers and other officials developing an error correction mechanism in order to have a more reliable and efficient supply system.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. H. Fredrickson	22b TELEPHONE (Include Area Code) 408-646-2235 22c. OFFICE SYMBOL 53FS

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

An Approach to the Concept of Error Recovery in the National Stock Number System

by

Mehmet Yenen
Lt. J.G., Turkish Navy
B.S., Turkish Naval Academy, 1980

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1986

ABSTRACT

This thesis pertains to the area of systematic error recovery for the National Supply System. By techniques applied to real stock numbers a correction algorithim is successfully developed. The main research area for this thesis is that of algebraic coding theory,especially Reed-Solomon codes and their application for the National Supply System using finite field theory and the RS code over GF(11).

The findings of the research are also discussed for a possible database interface. This product of the study may be used to assist supply officers and other officials developing an error correction mechanism in order to have a more reliable and efficient supply system.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

A. BACKGROUND

After World War II, a very important issue in the war was realized which, if used effectively, would bring the victory: Logistics.

One result, arising out of the second World War, has been the existence of two super powers in the world. Each of them has its own political, military and economic viewpoints. As a result of this, the NATO (North Atlantic Treaty Organization) has been established by the United States and some of the European countries who believe in the power and necessity of freedom. Since then, these members of NATO have worked together to provide required materials, military aids and education to each other, in order to develop a powerful and strong defense system against possible future threats.

During this development, a supply system has been needed to provide countries with required items in some designated time. The Supply System of the United States has been approved as the Supply System of NATO with some minor changes. Nowadays, most of the NATO members use this system as their own National Supply System.

However, from the establishment of NATO to the present, there has been no study about providing an error detection and correction mechanism in order to have a more reliable and efficient system.

The fundamental problem of communication is that of reproducing at one point, either exactly or approximately, a message selected at another point. The significant aspect is that the actual message is a selected message from a set of possible messages. The system must be designed to operate for each possible selection, not just the one which will actually be chosen [Ref. 1]. A National Stock Number (NSN),

can be considered as a message for this purposes. This thesis develops an error correction coding system for NSN's which allows correction of a broad class of errors.

B. GENERAL VIEW OF A SUPPLY SYSTEM

The existing supply system is a kind of bridge between resources and requirements by providing material at the required places at the required times. Therefore, the speed and accuracy of the supply are very important. In order to identify a specific item in the stock, it is necessary to provide an identifier and material classification.

1. Material Classification

There are over four million items in the United States Department of Defense Supply System. The Navy Supply System alone stocks over one million items. If we consider all NATO member's stocks, the size of the system would be staggering. For proper requisitioning of an item, a common language has been developed: the Federal Catalog System (FCS). The most important component of this system is the NSN.

2. National Stock Number (NSN)

A NSN is a 13 digit number assigned by the Defense Logistic Services Center (DLSC) to identify items of material in the Supply Distribution System of the United States. It consists of a four digit Federal Supply Classification Code (FSC), a two digit National Codification Bureau Code (BC), and a seven digit National Item Identification Number (NIIN). The NIIN part of a NSN is the most significant part and is used to uniquely identify each NSN item in the Federal Supply Distribution System [Ref. 2] Construction of a NSN can be seen in Figure 1.1.

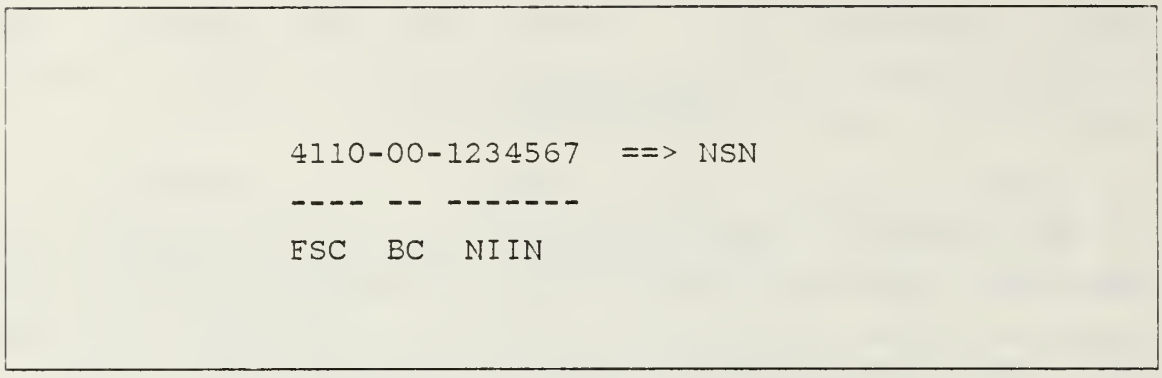


Figure 1.1 Construction of a NSN

3. Nato Stock Number

As mentioned earlier in this chapter, the main logic for the Nato Supply System is almost the same as in the National Supply System of the United States. The only difference is the names of the three parts existing in NSN. The corresponding names are explained in Table I.

The main difference is a two digit Country Code (also called Source Code) assigned for each NATO member instead of National Codification Bureau Code in the National Supply System. As an example, 00 and 01 represent the United States in this classification.

C. PURPOSE OF STUDY

The subject area of this thesis, the theory of error correcting codes, started as a subject in Electrical Engineering with Shannon's classic papers in 1948 and 1949. It has since become a mathematical topic and a part of the fascination has been the use of many varied tools to solve practical problems in coding. The possibility of applying finite field theory to problems in discrete communication was recognized in the late 1950's. One such class of codes which is very famous and popular in this area is the Reed-Solomon (RS) codes. These codes are chosen to built an error recovery mechanism in this thesis because of their effectiveness and ease of use.

The goal of this thesis is to provide an error recovery method for the existing Supply System using recent tools in the Computer Science area, such as data base applications, and combining them with the Reed-Solomon codes application.

In an effort to assist the reader in simplicity and comprehension of this abstract subject, this author has taken the pertinent information vital to the thesis and created a chapter for each. After the general view of the supply system is introduced in this chapter, the necessary fundamentals of finite fields and algebraic coding theory are presented in Chapters II and III. In Chapter IV the presentation of Reed-Solomon codes is made and the implementation theory is discussed in Chapter V. Implementation of the RS codes to the existing system is presented in Chapter VI. Finally, a possible interface between users and an existing database is described in Chapter VII using the implementation of Reed-Solomon codes discussed in Chapter VI. Conclusions of the thesis are represented in Chapter VIII.

TABLE I

CORRESPONDING NAMES IN NATO SUPPLY SYSTEM

Name Field in NSN	Corresponding in Nato Supply System
Federal Supply Classification (FSC)	Nato Supply Classification (NSC)
National Codifica. BUREAU Code (BC)	Nato Source (Country) Code (SC)
National Item Ident. Number (NIIN)	Nato Item Identification Number (NIIN)

II. ALGEBRAIC CODING THEORY

A. CODING WITH ALGEBRAS: LINEAR CODES

1. Vector Spaces

A vector space $V_n(K)$ of dimension n is a set of n long vectors $V = (v^1, v^2, v^3, \ldots, v^n)$ of elements in some field K and V forms an additive commutative group that also admits multiplication by scalars or elements from the field K. The rules for this scalar multiplication are;

If $a, b \in K$, $v_i \in V$ then

- $a*V \in V$; $1*v = v$; $a*(b*v) = (a*b)*v$
- $a*(v_1 + v_2) = a*v_1 + a*v_2$
- $(a + b)*v = a*v + b*v$

Addition of two vectors v_1 and v_2 is by component-wise addition by the addition defined in the field K. As an example, if $V_n(K)$ is the set of binary n-tuples, then K is the scalars, the elements 0 and 1 of the field of 2 elements.

2. Related Definitions

Linear Dependence: If V is a vector space and K is the scalar field, then a set of e vectors, $v_1, v_2, \ldots, v_e$, are said to be linearly dependent over K if there exists a set of scalars, $c_1, c_2, \ldots, c_e$, not all of them zero, such that;

$$c_1 * v_1 + c_2 * v_2 + \ldots + c_e * v_e = 0$$

Linear Independence: If the set of e vectors is not linearly dependent, i.e. , there do not exist scalars $c_1, c_2, \ldots, c_e$ such that

$$\sum_{i=i}^{e} c_i * v_i = 0$$

then the vectors are said to be linearly independent.

Dimension of V : Let m be the largest number of linearly independent vectors of V. The dimension of V is m.

B. GENERATOR MATRIX

Let A be an (n,k) group code. Choose k linearly independent vectors of A. Write these out as rows. The k rows form a (k*n) matrix G called the generator matrix for A. Every code vector is some linear combination of the rows of G.

This matrix description serves as a compact list of all code vectors. For example a (13,11) code can be described by a 11*13 t-ary matrix while the code contains t^{11} vectors where t is the size of the underlying field K.

1. Alternate Descriptions, Parity Check Matrix

If $a = (a_i)$ and $b = (b_i)$ then , we introduce the dot product (scalar product) of the two n-long vectors a and b, (a.b) as

$$(a.b) = \sum_{i=1}^{n} a_i.b_i \qquad (2.1)$$

It should be remembered that, sums and products are performed in K, which was introduced as the underlying field. If a.b = 0 then a is said to be orthogonal to b.

Now, let us consider a matrix H, which is a (n-k * n) matrix, whose rows are linearly independent.

Let V^1 be an orthogonal space to H, that is, a (V^1 --> $a.u^j = 0$, j=1,2,. . .,n-k where u^j is a row in V. In matrix notation $aH^T = 0$ (a is a 1*n vector here; T signifies transpose). H is called the parity check matrix to V^1. We first note that if a is orthogonal to the vectors (rows) of H, it is orthogonal to the vector space spanned by these vectors (by forming linear combinations of these vectors). We will illustrate by the following example how a parity check matrix can be obtained:

Example 3.1: Let H be defined as,

$$H = \begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{vmatrix}$$

In this example H is a (3*7) matrix. Then V^1 is a (7,4) group code containing the all one vector. We can also say that, the (13,11) code discussed in this section requires only a (2*13) parity check matrix for its description. An example of an important code obtainable using vector space or matrix methods is the Hamming code.

C. CODING WITH FINITE FIELDS: BINARY CYCLIC CODES

The most important group codes are the cyclic ones. These codes are distinguished by their ease of encoding and by the highly algebraic mechanizable approach to their decoding.

The first discovery in this area was made by Bose-Chauduri and Hocquenghem independently. But the approaches, introduced by these two, has minimal error correcting capability [Ref. 3]

Nowadays, there are many approaches to the exposition and treatment of these codes. However, certain mathematical knowledge and tools are required. For that reason, we introduce some more details before examining a most useful set of codes in this area , Reed-Solomon codes, in Chapter IV.

1. Finite Difference Equations

One of the most common approaches to cyclic codes is via finite difference equations. This approach represents one particular way of encoding that results in systematic cyclic codes. A major advantage of this approach is that, the encoding of cyclic codes becomes very natural and the decoding procedures emerge as a direct consequence.

Let us consider a finite difference equation of degree k, with constant coefficients in the field F:

$a_{n+k} + u_1 a_{n+k-1} + u_2 a_{n+k-2} + \ldots + u_{k-1} a_{n+1} + u_k a_n = 0$

where $u_i \in F$, $n = 0,1,2,3,\ldots$

We wish to solve the above difference equation, given k initial values; $a_0, a_1, a_2, \ldots, a_{k-1} \in F$. A general approach to the solution is illustrated in the following example.

Example 3.2: In this example we show that, the codeword (or output word) can be obtained by using the given finite difference equation and the given initial values. We say, $a_{n+3} + a_{n+1} + a_n = 0$ is the finite difference equation and $a_0 = 1$, $a_1 = 0$, $a_2 = 1$ are the initial values. The required operation could be performed over F, the field of 2 elements, in following way;

$a_3 = a_0 + a_1 = 1 + 0 = 1 \rightarrow a_3$

$a_4 = a_1 + a_2 = 0 + 1 = 1 \rightarrow a_4$

$a_5 = a_2 + a_3 = 1 + 1 = 0 \rightarrow a_5$

$a_6 = a_3 + a_4 = 1 + 1 = 0 \rightarrow a_6$

$a_7 = a_4 + a_5 = 1 + 0 = 1 \rightarrow a_0$

$a_8 = a_5 + a_6 = 0 + 0 = 0 \rightarrow a_1$

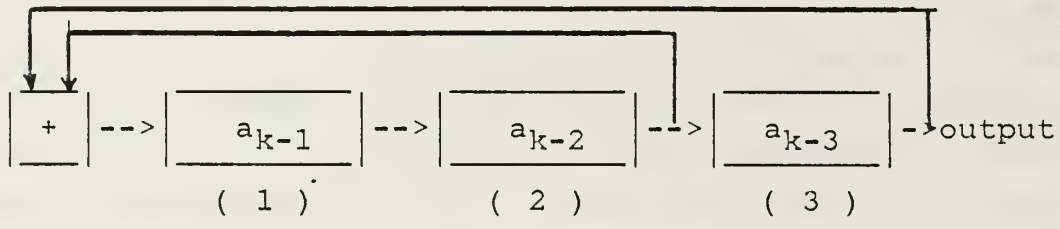
$a_9 = a_6 + a_7 = 0 + 1 = 1 \rightarrow a_2$

$a_{10} = a_7 + a_8 = 1 + 0 = 1 \rightarrow a_3$

and as it turns out, the sequences a_n obtained by this method are ultimately periodic.

Then the codeword obtained by performing this operation is 1011100 repeated periodically. We will use shift registers to provide the encoding process for this example. For the above problem, it is clear that, $a_{k-1} = a_{k-2} + a_{k-3}$ and we also know the initial values. All we need to do is,

construct a cycle in which the k^{th} element of the code will calculated just by adding previous register values. We use the given formula, when building the shift register diagram as represented in Figure 2.1.



STEP #:	POSITION #: #1	#2	#3	OUTPUT	
Initial	1	0	1	-	
1	1	1	0	1	---> a_0
2	1	1	1	0	---> a_1
3	0	1	1	1	---> a_2
4	0	0	1	1	---> a_3
5	1	0	0	1	---> a_4
6	0	1	0	0	---> a_5
7	1	0	1	0	---> a_6
8	1	1	0	1	---> a_0

CODEWORD (OUTPUT) : 1011100. . . .

Figure 2.1 Shift Register Application

D. (N,K) CYCLIC CODE

We have, via the recursion polynomial f(x), a natural mapping of k-tuples into n-tuples, given by the recursion rule (or difference equation):

$$a_0, a_1, \ldots, a_{k-1} \longrightarrow a_0, \ldots, a_{k-1}, a_k, \ldots, a_{n-1}$$

where the a_i are given by the difference equation. This mapping is linear, i.e., preserves the group structure. The number of distinct initial a's is 2^k and the linear property states, if a arises from $(a_0, a_1, \ldots, a_{k-1})$ and b from $(b_0, b_1, \ldots, b_{k-1})$, then a+b arises from $(a_0+b_0, a_1+b_1, \ldots, a_{k-1}+b_{k-1})$. Thus we have generated an (n,k) group code. This particular group code is cyclic in the sense that, if a = $(a_0, a_1, \ldots, a_{n-1})$ is a codeword, any word obtained by cyclically shifting a by a position to the right or left along this sequence is also a codeword. That is, $(a_1, a_2, \ldots, a_{n-1}, a_0)$, $(a_4, a_5, \ldots, a_0, a_1, a_2, a_3)$, etc. , are also codewords.

The conclusion is straightforward. If we choose any polynomial f(x) of degree k with coefficients in GF(2) and with no repeated roots, which divides x^{n+1}, but not x^{m+1}, m < n, then by forming the associated difference equation, we have a means of generating an (n,k) group code.

Example 3.3: Let $f(x) = x^3 + x + 1$. f(x) is a factor of $x^7 + 1$. The associated difference equation will give rise to a (7,3) cyclic code, where for example 111 -----> 1110010. Any other nonzero condition yields a codeword which is a cyclic shift of the given codeword.

Figure 2.2 shows how we produce this expansion. All we need to do is, sum up the last two digits, write down this sum as a new digit (to the left side), and shift all digits one position to the right. The codeword is obtained from first digit to seventh digit , i.e., from left to right.

Cyclic codes are useful for the ease of the encoding processes as they are easily mechanized by shift register devices. The cyclic property clearly minimizes storage facilities. Further these codes are easily analyzable, and also have very efficient decoding properties. The decoding

STEP #:	POSITION :	OUTPUT:	
Initial	111	-	C
Shift1	011	1	O
Shift2	001	1	D
Shift3	100	1	E
Shift4	010	0	W
Shift5	101	0	O
Shift6	110	1	R
Shift7	111	0	D

Figure 2.2 Creation of a Codeword

also proceeds via a shift register algorithm. We omit the details of the general shift register decoding here and illustrate the decoding in a specific instance later.

III. FINITE FIELD THEORY

A. BACKGROUND

Finite or Galois fields (named after the nineteenth century French mathematician Evariste Galois) play many important roles in signal processing and information theory applications. However, in this thesis we are concerned only with their use in the construction of Reed-Solomon error correcting codes. We begin with the general definitions in order to understand the pertinent facts regarding finite fields.

B. CONSTRUCTION

1. Definitions

A field is a set of elements, including 0 and 1, any pair of which may be added or multiplied, (denoted by + or *, respectively), to give a unique result in the field.

The basic building blocks are the prime fields F_p ,where p is a prime number. F_p is the field whose elements are 0,1,. . .,p-1, and arithmetic is performed modulo p. The additive structure is that of the vector space defined in the previous chapter.

The addition and multiplication are associative and commutative, and the multiplication distributes over addition in the usual way: u*(w+v)=u*w+u*v. Every field element u has a unique negative -u such that u+(-u)=0. Every nonzero field element u has a unique reciprocal field element 1/u, such that u*(1/u)=1. For every field element u, 0+u=u=1*u,and 0*u=0. Thus the numbers 0 and 1 are the additive and multiplicative identities, respectively.

The order of the field is the number of elements in the field. If the order is infinite, the field is called as an infinite field and if the number of elements is finite, we call the field a finite field.

C. MULTIPLICATIVE STRUCTURE

Let F_q be a finite field with $q = p^m$ elements. The nonzero elements of F_q form a commutative group, $(F_q)^*$, of order q-1, which is in fact a cyclic group under multiplication. Finite fields can be constructed as polynomial algebras by defining multiplication as polynomial multiplication. If we start with two polynomials f(x) and g(x) of degree less than n, then their product f(x)*g(x) when formed in the usual way is not necessarily a polynomial of degree less than n. In order to satisfy closure, we write [f(x)*g(x)] modulo P(x) where P(x) is an irreducible polynomial of degree n over the field. An element of multiplicative order q-1, that is, a generator of the group $(F_q)^*$, is called a primitive root.

It thus follows that every element a in $(F_q)^*$ satisfies $a^{q-1} = 1$, and so every element in F_q satisfies $a^q = a$. If F_q is viewed as a subfield of $F_q m$ for some m, then the equation characterizes F_q, that is, $a^q = a$ iff a is an element of F_q. In other words, the multiplicative group of nonzero field elements is cyclic, i.e., it is a group that consists of all the powers of one of its elements, a. Multiplication can alternatively be defined as $a^i * a^j = a^{i+j}$ where i+j is computed modulo (p^m-1) and a is the generator of this group.

D. THE MINIMAL POLYNOMIAL

The minimal polynomial of a is defined to be the monic polynomial f(x) of least degree with coefficients in F_p such that f(a) = 0. Over F_p f(x) is irreducible, but in the larger field F_q f(x) factors into linear factors:

$$f(x)=(x-a)(x-a^p)\ldots(x-a^{p^{k-1}}) \qquad (3.1)$$

where k is called the degree of a and thus the degree of f(x) is the same as the degree of a.

If a is a primitive root in $F_p m$, the minimal polynomial of a is called a primitive polynomial over F_p. It is often convenient to reverse this process and use a primitive polynomial to construct a field.

1. An Example Of The Creation Of a Field

Consider the Galois field GF(2^4). It has 2^4 elements and may be constructed as the field of polynomials over GF(2) modulo the irreducible polynomial $1= x + x^4$. If we let b represent a root of this irreducible polynomial, then it is also a primitive element of the field. Field addition of the elements is bit-by-bit modulo 2 addition while multiplication of the elements is described using the primitivity of the element b. Thus, $b^i * b^j = b^{i+j}$ where i+j is reduced modulo 15, if necessary.

After defining the addition and multiplication over GF(2^4), we are ready to create the field. First, we need to have a primitive polynomial which will be order of 4. We will select the primitive polynomial as $f(x)= x^4+x+1$. The operations which are required to create the field elements can be performed as :

$f(b)= 0 \Rightarrow b^4 = b + 1$

$b^5 = b^2 + b$

$b^6 = b^3 + b^2$

$b^7 = b^3 + b + 1$

- - - - - -

- - - - - -

$b^{14} = b^3 + 1$

$b^{15} = 1$

$b^{16} = b$

$b^{17} = b^2$

$$b^{18} = b^3$$

$$b^{19} = b^{16} + b^{15} = b + 1 = b^4$$

The field elements are listed in Table II.

TABLE II

REPRESENTATION OF GF(2^4)

Field element	b Polynomial	4-Tuple
0	- - - - - -	0000
b^0	1	0001
b^1	b	0010
b^2	b^2	0100
b^3	b^3	1000
b^4	b +1	0011
b^5	b^2+ b	0110
b^6	b^3 + b^2	1100
b^7	b^3 + b +1	1011
b^8	b^2 +1	0101
b^9	b^3 + b	1010
b^{10}	b^2+ b +1	0111
b^{11}	b^3 + b^2+ b	1110
b^{12}	b^3 + b^2+ b +1	1111
b^{13}	b^3 + b^2 +1	1101
b^{14}	b^3 +1	1001

IV. REED-SOLOMON CODES

A. BACKGROUND

Cyclic codes over an alphabet of p^m symbols were originally introduced by Gorenstein and Zierler (1961) along with an effective error correcting procedure. They also pointed out that the Reed-Solomon burst error correcting code may be considered as a code in this class [Ref. 4]

Reed-Solomon codes are (q^n-1,m) cyclic codes over $GF(q^n)$ and were originally defined by Reed-Solomon differently than the Gorenstein and Zierler formulations. The code RS(n,t) is called a k error correcting Reed-Solomon code of length n. In this definition t is the number of information symbols in the codeword and $k < n-t$. These codes can correct both random and burst errors over a communication channel and hence are ideal for the numerous real time and reliable communications demanded by these applications. The complexity of RS encoders and decoders are proportional to the error correcting capability of the code, the speed of the decoding and the interleaving depth used.

The code RS(n,t) consists of all vectors $C = (C_0, C_1, \ldots, C_{n-1}) \in V_n(Fq^m)$ such that the corresponding polynomial $C(x) = \sum_{i=0}^{n-1} C^i x^i$ has the form $C(x) = (x - a)(x - a^2) \ldots (x - a^{2t}) I(x)$, where I(x) is a polynomial of degree $< n-1-2t$ over Fq^m. I(x) is the polynomial of the information symbols while $(x - a)(x - a^2) \ldots (x - a^{2t})$ is the polynomial of the check symbols. The code has parameters:

Length	:	$n = q^m-1$
Dimension	:	$k = n - 2t$
Min. distance:		$d = 2t + 1$

B. GENERAL ENCODING PROCESS

As mentioned earlier, Reed-Solomon codes are cyclic codes over $GF(q^n)$. Let $a_0, a_1, \ldots, a_{m-1}$ be elements of $GF(q^n)$, then the code is defined in a non-systematic way as follows:

$$a(x) = a_0 + a_1x + \ldots + a_{m-1}x^{m-1} \qquad (4.1)$$

We let c be a $(q^n-1)^{th}$ root of unity. We define $b = (b_0, b_1, \ldots, b_{q^n-2})$ to be the vector whose coordinates are given by $b_i = a(c^i)$, $i = 0, 1, \ldots, q^{n-2}$. The code map $(a_0, a_2, \ldots, a_{m-1}) \dashrightarrow (b_0, b_1, \ldots, b_{q^n-2})$ gives rise to a (q^n-1, m) code with maximum distance $d > q^n - 1 - (m-1) = q^n - m$ which will correct errors up to

$$e < (q^n - m)/ 2 \qquad (4.2)$$

Here the length of the codeword is $q^n - 1$ and the field in which the symbols lie is $GF(q^n)$. For any vector a in $V_{q^n-1}(q^n)$, the codeword associated with a(x) (ga(x) is written very simply;

$$ga(x)=c_0 + c_1x + \ldots + (c_{q^n-1})x^{q^n-2} \qquad (4.3)$$

The recursion rule or polynomial associated with the code is;

$$\prod_{i=1}^{m-1} (x - c^i) = f(x) \qquad (4.4)$$

The Reed-Solomon code may thus be encoded systematically.

1. General Encoding Algorithm

As discussed in Chapter III, an (n,k) code can be generated by a polynomial of degree n-k. If the polynomial

is primitive of degree r and $n = 2^r-1$, the code can be encoded and decoded with primitive shift registers [Ref. 5]. Hence, we restrict our attention solely to the case of primitive polynomials. We illustrate the required algorithm for encoding of RS codes using primitive polynomials as follows:

1. Represent the message as a polynomial. Call this polynomial m(x). Degree of m(x) is at most k.
2. Multiply m(x) by x^{n-k} to shift the message digits to the far right.
3. Calculate the remainder when x^{n-k} m(x) is divided by p(x). (Note that p(x) is the primitive polynomial).
4. Form the code polynomial as the sum x^{n-k} m(x) + r(x), (r(x) is the remainder of the division). The check polynomial is actually then a multiple of p(x).

C. GENERAL DECODING PROCESS

The problem for decoding is to find the error positions and symbol changes. There are two decoding procedures available to do this. The first of them requires finding the coefficient matrix which will determine the symmetric functions of the error positions, @, or finding the appropriate augmented matrix and computing the syndromes, (S_j), for $j > d_0$. The second method will not be addressed in this thesis.

1. General Decoding Algorithm

The decoding of an RS code is composed of the following six steps. For this algorithm, the input is the received vector R and the output is the codeword C.

1. Compute the syndromes, using the equations:

$$S_j = \sum_{i=0}^{n-1} R_i a^{ij} \quad , \quad j = 0,1,2. \ . \ .2t-1. \qquad (4.5)$$

2. Perform Euclid's algorithm on x^{2t} and $S(x) = S_1 + S_2x + \ . \ . \ . \ + S_{2t}x^{2t-1}$. Stop as soon as the degree of the remainder $r_j < t$. Use the same algorithm to determine

the coefficients of the error locator polynomial $@(x)$. Then calculate $w(x)$, the error evaluator polynomial, using the fact, $@(x).S(x) = w(x) \pmod{g(x)}$, where $g(x)$ is an irreducible polynomial over the field.

3. Find $B = \{ b \in F_q m : @(b) = 0 \}$, by trial and error method.
4. For each $b \in B$, set $E_b = w(b)/@'(b)$ where $@'(b)$ is the formal derivative of $@(b)$.
5. For each $i = 0,1,\ldots,n-1$, set

$$E_i = \begin{cases} 0 & \text{if } a^{-i} \notin B, \\ E_b & \text{if } a^{-i} \in B. \end{cases}$$

6. Output the codeword by subtracting the error vector E_i from the received vector R_i;

$$C = (R_0-E_0 , R_1-E_1 , \ldots , R_{n-1}-E_{n-1}).$$

V. IMPLEMENTATION THEORY

A. BACKGROUND

In this chapter we look at the theoretical concepts behind the creation of a particular finite field and construction of a specific RS code. As mentioned in Chapter I, a NSN (National Stock Number) consists of three basic units and one of them is NIIN (National Item Identification Number). It has also mentioned that the NIIN part of a stock number is unique for each supply item. Thus, we will consider only the NIIN part of the stock number when we construct our coding scheme in the subsequent chapter.

There are seven digits in the NIIN part of the stock number. These symbols will be the information symbols in the codeword. We also, in this chapter, discuss the number and the types of the errors we consider for the correction process. That is, we decide how many error check digits we must have in the codeword. We also introduce the order and construction of the finite field we use for our particular application. Since the symbols are not binary, the Reed-Solomon codeword we will design contains symbols which lie in a larger field than GF(2). In particular, since the symbols are digits, the field will have to have at least 10 elements. As the size of field must be a power of a prime number, we shall use a field of 11 elements.

We also, in this chapter, discuss and decide upon the generating polynomial g(x) and find a primitive root of unity (generator), a, for our construction of the Reed-Solomon code.

B. ERROR CHECK DIGITS

Before going further with the discussion and construction of the finite field we shall need, it is necessary to

find out the number of error check digits which will be used in the codeword. Our coding scheme will be an interface between users and computers. In order to figure out the number of check symbols, we consider the common type of errors which are most likely to be made by humans. When dealing with humans, three types of errors are common [Ref. 6]

1. People have a tendency to interchange adjacent digits of numbers; for example 67 becomes 76.
2. Another common error is to double the wrong one of a triple of digits, two adjacent ones of which are the same; for example 667 becomes 677 merely by a change of one digit.
3. A third kind of simple error is just the substitution of one symbol for another.

These are the most common errors in arithmetic. We will provide for the correction of two errors in our application. This will handle the first one of these common errors as well as the other errors mentioned. So, the maximum distance required as discussed previously can be found by applying the equation;

$$d_0 > q^n - m = q^n - 7 \qquad (5.1)$$

where m represents the number of information symbols in the codeword. In the NIIN portion of the message, there are 7 digits present.

Since we want to correct up to two errors, according to the equation 4.2 which we introduced in Chapter IV;

$$e \leq q^n/2 - m/2 \geq 2 \Longrightarrow q^n \geq 11 \qquad (5.2)$$

Accordingly we have found from the above equation, we will choose GF(11) for our application. Here, q = 11 and n = 1, and the computations will be in arithmetic modulo 11. To make this approach clear, we illustrate addition and multiplication over GF(11) in Table III and Table IV respectively. The symbol A is used to represent the digit 10.

Our findings from equation 4.2 show that, we will have four error check digits in the codeword. One of these (the last one on the right) is the parity check digit. Parity checking for this type of coding scheme provides the total error amount occuring in the codeword computed modulo 11. It is obvious that, if this error amount is zero, there are several possible cases to be considered. Either no error occured, or there are some interchanged bits in the codeword, or the total error amount is a multiple of 11. Therefore we provide a decoding process by calculating all the syndromes first. Then the error correction procedure is as described in the decoding procedure of Chapter IV and more precisely in Chapter VI.

C. GENERATING POLYNOMIAL

As mentioned in the previous chapter, the generating polynomial for the RS code is described by the equation:

$$g(x) = (x - b)(x - b^2). \ . \ .(x - b^{2t-1}) \qquad (5.3)$$

where t is the error correcting capability of the RS code and b is a primitive root of unity (generator). The significant point here is to find the generator element first in order to determine the generating polynomial according to equation 5.3. Since the RS code is cyclic; the powers of a primitive root should generate all the roots of unity. In other words, the powers of the generator should generate all the nonzero elements of the finite field, namely GF(11).

It can be determined that 2 is a generator, by trial and error method. For GF(11) the verification that 2 is a generator is shown in Figure 5.1. Since we have the generator 2 and we have already decided the error correcting capability of the code, (2 error correction) we are now able to calculate the generating polynomial.

TABLE III

ADDITION OVER GF(11)

+	0	1	2	3	4	5	6	7	8	9	A
0	0	1	2	3	4	5	6	7	8	9	A
1	1	2	3	4	5	6	7	8	9	A	0
2	2	3	4	5	6	7	8	9	A	0	1
3	3	4	5	6	7	8	9	A	0	1	2
4	4	5	6	7	8	9	A	0	1	2	3
5	5	6	7	8	9	A	0	1	2	3	4
6	6	7	8	9	A	0	1	2	3	4	5
7	7	8	9	A	0	1	2	3	4	5	6
8	8	9	A	0	1	2	3	4	5	6	7
9	9	A	0	1	2	3	4	5	6	7	8
A	A	0	1	2	3	4	5	6	7	8	9

TABLE IV

MULTIPLICATION OVER GF(11)

*	0	1	2	3	4	5	6	7	8	9	A
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A
2	0	2	4	6	8	A	1	3	5	7	9
3	0	3	6	9	1	4	7	A	2	5	8
4	0	4	8	1	5	9	2	6	A	3	7
5	0	5	A	4	9	3	8	2	7	1	6
6	0	6	1	7	2	8	3	9	4	A	5
7	0	7	3	A	6	2	9	5	1	8	4
8	0	8	5	2	A	7	4	1	9	6	3
9	0	9	7	5	3	1	A	8	6	4	2
A	0	A	9	8	7	6	5	4	3	2	1

The generating polynomial can be obtained with reference to the equation 5.3 and Figure 5.1 in the following way:

$$g(x) = (x - b^1)(x - b^2)(x - b^3)$$

$$= (x - 2)(x - 4)(x - 8)$$

$$= (x^2 - 6x + 8)(x - 8)$$

So, the generating polynomial is described by the equation;

$$g(x) = x^3 - 3x^2 + x + 2 \qquad (5.4)$$

where sums and products are computed modulo 11.

b	9	8	7	6	5	4	3	2	1	0
2^b	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
VALUE	6	3	7	9	A	5	8	4	2	1

Figure 5.1 Verification of the generator b = 2

D. DECODING TOOLS

1. Syndromes

In chapter IV, we introduced the general syndrome calculation for the Reed-Solomon code by the equation;

$$S_j = \sum_{i=0}^{n-1} R_i a^{-ji} , \quad j = 0,1,2...2t-1 \qquad (5.5)$$

where a represents the primitive root of unity (generator) and t represents the error correcting capability of the RS code. Referring to equation 5.5, we will have four syndromes namely S_0, S_1, S_2, S_3. The first syndrome, S_0, will show us the error amount occuring in the codeword. The last digit of the codeword, the one we appended for parity checking,will only be used in calculating the syndrome value S_0. We will not use the last digit for calculating the other three syndromes S_1, S_2 and S_3.

2. Conditions of Syndromes in Case of Errors

After calculating the syndromes, we will be able to examine the received codeword and make some decisions about it. From the syndromes S_0, S_1, S_2 and S_3 it is possible to find and correct up to and including any two symbol errors according to the following descriptions.

In case of no error, all the syndromes will be equal to zero.

$$S_0 = S_1 = S_2 = S_3 = 0 \qquad (5.6)$$

In case of one error, the syndromes have the property of

$$S_0 = e_k$$

$$S_1 = e_k a^k = e_k 2^k$$

$$S_2 = e_k a^{2k} = e_k 2^{2k}$$

$$S_3 = e_k a^{3k} = e_k 2^{3k}$$

where e_k is the error amount occuring in the k^{th} digit. So, the equality we will be looking for in case of one error is

$$S_1/S_0 = S_2/S_1 = S_3/S_2 = 2^k \qquad (5.7)$$

Thus the location of the error is determined by $S_1/S_0 = 2^k$, the k^{th} digit and the amount of the error is $S_0 = e_k$.

In case of two errors, the syndromes are given by

$$S_0 = e_k + e_l$$

$$S_1 = e_k a^k + e_l a^l$$

$$S_2 = e_k a^{2k} + e_l a^{2l}$$

$$S_3 = e_k a^{3k} + e_l a^{3l}$$

where the two errors are e_k and e_l occuring in the k^{th} and l^{th} digit of the codeword, respectively. The inequality we will consider in case of two errors is

$$S_1/S_0 \neq S_2/S_1 \neq S_3/S_2 \qquad (5.8)$$

That is, not all of S_1/S_0, S_2/S_1 and S_3/S_2 are equal. In the next section we give the procedure to determine the locations of the errors in case of two errors.

E. DECIDING ON THE POSITIONS OF ERRORS

Once the syndromes are calculated, we are able to decide on the positions of errors occuring (if any). But, it is required to have an equation to do that which will have only l and k (the positions of errors) as unknowns. Such an equation can be obtained in the following way:

The syndromes have been determined as

$$S_0 = e + d$$

$$S_1 = e2^k + d2^l$$

$$S_2 = e2^{2k} + d2^{2l}$$

$$S_3 = e2^{3k} + d2^{3l}$$

where e and d represent the error amounts and k and l represent the error positions, respectively.

Now, we multiply the syndromes S_1 and S_3 forming

$$S_1S_3 = e^2 2^{4k} + d^2 2^{4l} + ed(2^k 2^{3l} + 2^l 2^{3k})$$

$$= e^2 2^{4k} + d^2 2^{4l} + ed2^{k+l}(2^{2l} + 2^{2k})$$

Next we form the square of the syndrome S_2;

$$(S_2)^2 = e^2 2^{4k} + d^2 2^{4l} + 2ed2^{k+l}(2^{k+l})$$

When we subtract $(S_2)^2$ from S_1S_3 we obtain

$$S_1S_3 - (S_2)^2 = ed2^{k+l}(-2(2^k 2^l) + 2^{2l} + 2^{2k})$$

And finally we get the equation

$$S_1S_3 - S_2^2 = ed2^k 2^l (2^l - 2^k)^2 \qquad (5.9)$$

In the following steps, we substitute e and d in equation 5.9, so that we get an equation including only k and l as unknowns. First, by referring to the first syndrome S_0, it can be derived that, $e = S_0 - d$. When we substitute this value of e in the second syndrome equation, we get

$$(S_0 - d)2^k + d2^l - S_1 = 0$$

$$d(2^l - 2^k) + S_0 2^k - S_1 = 0$$

So, the value of d can be obtained as

$$d = (S_1 - 2^k S_0) / (2^l - 2^k) \qquad (5.10)$$

Now, we substitute the value of d obtained in equation 5.10, into equation 5.9 yielding

$$S_1S_3 - (S_2)^2 = e\left[(S_1 - S_0 2^k) / (2^l - 2^k)\right] 2^k 2^l (2^l - 2^k)^2$$

$$= e(S_1 - S_0 2^k)\, 2^k 2^l (2^l - 2^k)$$

Now using again the equation, $e = S_0 - d$, the value of e can be calculated as;

$$e = (S_0 2^l - S_1) / (2^l - 2^k) \qquad (5.11)$$

Now, we substitute equation 5.11 into equation 5.9;

$$S_1S_3 - (S_2)^2 = (S_0 2^l - S_1)(S_1 - S_0 2^k) 2^l 2^k$$

$$= (S_0 2^l - S_1)\, 2^l\, (S_1 - S_0 2^k)\, 2^k$$

Finally, we get the equation;

$$S_1S_3 - S^2_2 = (S_0 2^{2l} - S_1 2^l)(S_1 2^k - S_0 2^{2k})$$

As seen in the above equation, the only unknown terms are, the error positions k and l. We use this equation to locate the error positions by a trial and error procedure in the subsequent chapter.

VI. IMPLEMENTATION

A. BACKGROUND

The fundamental and necessary concepts for constructing an RS(11,7) code are discussed in the previous chapter. Our findings from Chapter V are used in this chapter to implement the RS(11,7) code. We use the generating polynomial, primitive root of unity, syndromes and the relationship between the positions of the errors occuring as we found and discussed them in Chapter V.

We also, in this chapter, provide the encoding and decoding algorithms in more detail and also illustrate some examples to show their application.

B. ENCODING PROCESS

As discussed in chapter IV, the RS codewords are formed as multiples of the primitive generating polynomial $g(x)$. As $g(x)$ is of degree r, there are $n - r = k$ information symbols which can be chosen freely. Then r check symbols are determined so that the resulting codeword satisfies the criteria stated, namely that the codewords are multiples of the generator polynomial. In other words, the check digits are the coefficients of the remainder $r(x)$ upon division of the information polynomial $p(x)$ by $g(x)$ as shown in example 6.1. Here we consider n as 10, as our codeword is of length 10. We also append the parity check digit after we calculate the first three check digits. So $p(x)$ can be obtained as a polynomial of degree 9 and a parity check symbol appended.

1. Encoding Algorithm

1. Represent the NIIN part of the stock number as coefficients of the polynomial of degree 9. Call this polynomial $p(x)$. Thus x^3 * (the polynomial of degree 6 representing the 7 digits of NIIN) is the representation.

2. Perform the required division $p(x)/g(x)$ where $g(x)$ is the generating polynomial having the value of $x^3 - 3x^2 + x + 2$.

3. Calculate the check digits as the coefficients of the remainder $r(x)$ upon division of $p(x)/g(x)$ and add these check digits to the right of the NIIN.

4. Calculate the parity check digit, by using the equation;

$$\sum_{i=0}^{9} C_i + p = 0 \pmod{11} \qquad (6.1)$$

where p represents the parity check digit value and C vector of first 10 digits of the codeword.

5. Append the parity check digit to the far right and output the 11 digit number as the encoded codeword C.

Example 6.1 : Now we will give an example of the encoding process by applying the encoding algorithm we developed. Suppose the NIIN part of the stock number is 0000001. If we represent it as the coefficients of the $p(x)$, of degree 9, we obtain $p(x) = x^3$, and we satisfy step 1 of the algorithm. According to step 2, we perform the required division of $p(x)/g(x)$ and get the remainder $r(x) = 3x^2 - x - 2$. So, this will give us the first three check digits as -3, 1 and 2. Now we add these digits to the right of NIIN and get C = 0000001-312. Then applying the equation 6.1, we calculate the parity check digit p as -1, so that $\sum_{i=0}^{9} C_i + p = 0$. Appending this parity check digit to the far right, yields the encoded codeword 0000001-312-1. We use this encoded codeword in our decoding examples in this chapter and call it C. When C is received, with or without errors, it is called R.

Example 6.2 : We now provide another example to show how the required division is performed in a more complex case. Suppose the NIIN part of the stock number is 9876543. The required division p(x)/g(x) can be seen in Figure 6.1. Then, the check digits are obtained as A, 8, A (A represents number 10). Applying the encoding algorithm step 4, the parity check digit will be obtained as 7. So the output of the algorithm is the codeword R = 9876543A8A7.

C. DECODING PROCESS

The decoding process is, in general, much more complicated than the encoding process. Not only must we deal with the detection of errors but also with their correction. Error detection is much easier than error correction. Since the code polynomial is a multiple of the generating polynomial p(x), the received code polynomial R(x) will be a code polynomial if and only if the remainder upon division of R(x) by p(x) is zero.

There is only one condition for a valid codeword, that is the equality of all the syndromes to zero. This condition is the desired one we will be trying to satisfy throughout the entire decoding process.

We will develop our decoding algorithm, considering the three possible conditions which are related with the errors occuring in the encoded codeword. They are listed below:

1. No error condition
2. One error condition
3. Two errors condition

For the first condition, we have already shown that, all the syndromes are equal to zero. So, it is easy to determine the no error condition. The decoded (output) codeword will be the same as the received codeword. In case of one error, it is necessary to figure out the error position first. We

$$
\begin{array}{r|l}
 & 9x^6 + 2x^5 + 4x^4 + 9x^3 + 2x^2 + 4x + 6 \\
\hline
x^3+8x^2+x+2 & 9x^9+8x^8+7x^7+6x^6+5x^5+4x^4+3x^3 \\
 & -9x^9-6x^8-9x^7-7x^6 \\
\hline
 & 2x^8+9x^7+Ax^6+5x^5 \\
 & -2x^8-5x^7-2x^6-4x^5 \\
\hline
 & 4x^7+8x^6+x^5+4x^4 \\
 & -4x^7-Ax^6-4x_5-8x^4 \\
\hline
 & 9x^6+8x^5+7x^4+3x^3 \\
 & -9x^6-6x^5-9x^4-7x^3 \\
\hline
 & 2x^5+9x^4+7x^3+0x^2 \\
 & -2x^5-5x^4-2x^3-4x^2 \\
\hline
 & 4x^4+5x^3+7x^2+0x \\
 & -4x^4-Ax^3-4x^2-8x \\
\hline
 & 6x^3+3x^2+3x+0 \\
 & -6x^3-4x^2-6x-1 \\
\hline
 & Ax^2+8x+A
\end{array}
$$

Since the coefficients of the remainder will be the values of the first three error check digits, they will be A, 8 and A.

Figure 6.1 Required Division for Example 6.2

will be using the equality we described for the first syndrome S_1 to decide the error position in our decoding algorithm.

Once the syndrome S_1 is calculated from the equation $S_1 = \sum_{j=0}^{n-1} R_j 2^{-j}$ we can find out the error position using the other equation described for calculating the first syndrome, $S_1 = R_k e^k$, where $R_k = S_0$ and it is known. After deciding the error position, the necessary correction process is quite simple, as the actual error is equal to S_0, and it will be explained in the decoding algorithm.

In case of two errors, the correction process is much more complex than the other two cases. Again, we must decide the error positions first. We use the equation described in step 10 of the decoding algorithm to do that. After finding out the error positions by trial and error method and the equation of step 10, we use two different tables to determine the error amounts occuring in the k^{th} and l^{th} digits of the codeword, the determined error location points. The way to use these two tables will also be explained in the decoding algorithm.

We follow a systematic procedure to describe the steps of the algorithm so that, steps number two through four are related to the no error condition, steps number five through eight are related to the one error condition and steps number nine through fifteen handle the two error condition. Step number one, sixteen, seventeen and eighteen are the common steps and they are used each time the decoding algorithm is applied. Note: if it is determined that none of the three possibilities is operable then we say that more than 2 errors have occured and a decoding failure is declared.

1. Decoding Algorithm

1. Compute the syndromes from the received word R:

$$S_i = \sum_{j=0}^{n-1} R_j 2^{-ji}$$

2. If $S_0 = S_1 = S_2 = S_3 = 0$, then decide "NO ERROR".

3. Set error vector E to all 0's.

4. Go to step 16.

5. If $S_1/S_0 = S_2/S_1 = S_3/S_2$ then decide "ONE ERROR".

6. Calculate k, the error position, using the equation;

$$S_1/S_0 = k$$

7. Set error vector E to;

$$E = \begin{cases} 0 & E_i \neq E_k \\ S_0 & E_i = E_k \end{cases} \quad \text{where } i = 0,1,\ldots,10$$

8. Goto step 16.

9. If not all of S_1/S_0 , S_2/S_1 , S_3/S_2 are equal then decide "TWO ERRORS".

10. Decide l and k, which correspond to the locations of errors occuring, using the following equation

$$S_1S_3-(S_2)^2=(S_0 2^{2(9-l)}+S_1 2^{9-l})(S_1\ 2^{9-k}+S_0 2^{2(9-k)})$$

where k and l lie in the range of 0 to 9.

11. Set i to 1 which corresponds to the index of of Table V.

12. Pick the i^{th} pair of e,d which corresponds to the error amount occuring using Table V.

13. Try to satisfy the equation using the present values of e, d, k, l;

$$S_1 = e2^{9-k} + d2^{9-l}$$

14. If the equation is not satisfied,then increment i by 1 and go to step 12.
If none of the values of i allow a solution then declare that more than 2 errors occured occured and a decoding failure results.

15. Set error vector E to;

$$E = \begin{cases} e & E_i = E_k \\ d & E_i = E_l \text{ where } i = 0,1,\ldots,10 \\ 0 & \text{otherwise} \end{cases}$$

16. Output the codeword by subtracting the error vector from received vector.

$$C = R - E = (R_0-E_0,..,R_{10}-E_{10})$$

17. Check syndromes again and verify that;

$$S_0 = S_1 = S_2 = S_3 = 0.$$

18. Then the information symbols are :

$$C_0, C_1,\ldots,C_6.$$

x	0	1	2	3	4	5	6	7	8	9
2^{9-x}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
VALUE	6	3	7	9	A	5	8	4	2	1

Figure 6.2 Values of the 2^{9-x}

TABLE V

POSSIBLE VALUES OF THE PARAMETERS E AND D

i	S = 0		S = 1		S = 2		S = 3		S = 4		S = 5		S = 6		S = 7		S = 8		S = 9		S = 10	
	e	d	e	d	e	d	e	d	e	d	e	d	e	d	e	d	e	d	e	d	e	d
1	1	10	-----		1	1	1	2	1	3	1	4	1	5	1	6	1	7	1	8	1	9
2	2	9	2	10	-----		2	1	2	2	2	3	2	4	2	5	2	6	2	7	2	8
3	3	8	3	9	3	10	-----		3	1	3	2	3	3	3	4	3	5	3	6	3	7
4	4	7	4	8	4	9	4	10	-----		4	1	4	2	4	3	4	4	4	5	4	6
5	5	6	5	7	5	8	5	9	5	10	------		5	1	5	2	5	3	5	4	5	5
6	6	5	6	6	6	7	6	8	6	9	6	10	-----		6	1	6	2	6	3	6	4
7	7	4	7	5	7	6	7	7	7	8	7	9	7	10	------		7	1	7	2	7	3
8	8	3	8	4	8	5	8	6	8	7	8	8	8	9	8	10	-----		8	1	8	2
9	9	2	9	3	9	4	9	5	9	6	9	7	9	8	9	9	9	10	------		9	1
10	10	1	10	2	10	3	10	4	10	5	10	6	10	7	10	8	10	9	10	10	-----	

We illustrate the decoding algorithm we presented in this chapter by three examples. Each of these examples will correspond to one of the error conditions we discussed in the decoding algorithm.

We use the received codeword R, as mentioned in example 6.2 throughout the examples we will present in this section. In each case, the value of C is 0000001-312-1.

We will first begin with the no error condition:

Example 6.3 : Suppose we have received the codeword R = 0000001-312-1. Referring to the decoding algorithm, first we calculate the syndromes in the following way:

Received Codeword	0	0	0	0	0	0	1	-3	1	2	-1
Position # (k)	0	1	2	3	4	5	6	7	8	9	10

Applying the decoding algorithm step 1, the syndromes will be;

$$S_0 = 1 - 3 + 1 + 2 - 1 = 0.$$

S_1 ===>

1	-3	1	2
2^3	2^2	2^1	2^0
8	-1	2	2

====> $S_1 = 8 - 1 + 2 + 2 = 0.$

S_2 ===> 1 -3 1 2

2^6 2^4 2^2 2^0

---

9 -4 4 2 ====> S_2 = 9 - 4 + 4 + 2 = 0.

S_3 ===> 1 -3 1 2

2^9 2^6 2^3 2^0

---

6 -5 8 2 ====> S_3 = 6 - 5 + 8 + 2 = 0.

Since $S_0 = S_1 = S_2 = S_3 = 0$, decide "NO ERROR".

Set the error vector E to all zeros;

E = 00000000000

Applying step 16 of the algorithm;

R = 0 0 0 0 0 0 1 -3 1 2 -1

\- E = 0 0 0 0 0 0 0 0 0 0 0

C = 0 0 0 0 0 0 1 -3 1 2 -1

Check and verify the syndromes and output the verified codeword as C = 0000001-312-1. The information is then determined as 0000001.

Example 6.4 : Suppose we have received the codeword R = 0000011-312-1. Referring to the decoding algorithm; first we will calculate the syndromes in the same way as in the previous example.

Received Codeword	0	0	0	0	0	1	1	-3	1	2	-1
Position # (k)	0	1	2	3	4	5	6	7	8	9	10

Applying the decoding algorithm step 1, the syndromes will be;

$$S_0 = 1 + 1 - 3 + 1 + 2 - 1 = 1.$$

S_1 ===> 1 1 -3 1 2
2^4 2^3 2^2 2^1 2^0

5 8 -1 2 2 ===> $S_1 = 5 + 8 - 1 + 2 + 2 = 5.$

S_2 ===> 1 1 -3 1 2
2^8 2^6 2^4 2^2 2^0

3 9 -4 4 2 ===> $S_2 = 3 + 9 - 4 + 4 + 2 = 3.$

S_3 ===> 1 1 -3 1 2
2^2 2^9 2^6 2^3 2^0

4 6 -5 8 2 ===> $S_3 = 4 + 6 - 5 + 8 + 2 = 4.$

The syndromes satisfy the condition;

$S_1/S_0 = S_2/S_1 = S_3/S_2 = 2^4$. Decide "ONE ERROR".

According to the decoding algorithm step 6;

$S_0 = E_k = 1$

$S_1 = 5 = 2^4 = E_k 2^{9-k}$ ===> $k = 5$.

So, decide the error position as position number 5 and the error as 1.

Set the error vector E as described in step 7 of the algorithm;

E = 00000100000

Applying step 16 of the algorithm;

R = 0 0 0 0 0 1 1 -3 1 2 -1

\- E = 0 0 0 0 0 1 0 0 0 0 0

C = 0 0 0 0 0 0 1 -3 1 2 -1

Check and verify the syndromes and output the verified codeword as C = 0000001-312-1. Again the information is 0000001.

Example 6.5 : Suppose we have received the codeword R = 0001101-312-1. Referring to the decoding algorithm, we again calculate the syndromes in the usual way.

Received Codeword	0	0	0	1	1	0	1	-3	1	2	-1
Position # (k)	0	1	2	3	4	5	6	7	8	9	10

Applying the decoding algorithm step 1, the syndromes will be;

$$S_0 = 1 + 1 + 1 - 3 + 1 + 2 - 1 = 2$$

S_1 ===> 1 1 0 1 -3 1 2
$2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

9 -1 0 8 -1 2 2 ===> $S_1 = 8 = 2^3$

S_2 ===> 1 1 0 1 -3 1 2
$2^2\ 2^0\ 2^8\ 2^6\ 2^4\ 2^2\ 2^0$

4 -1 0 9 -4 4 2 ===> $S_2 = 3 = 2^8$

S_3 ===> 1 1 0 1 -3 1 2
$2^8\ 2^5\ 2^2\ 2^9\ 2^6\ 2^3\ 2^0$

3 -1 0 6 -5 8 2 ===> $S_3 = 2$

Since the syndromes are not all zero and do not satisfy the condition;

$S_1/S_0 = S_2/S_1 = S_3/S_2$, decide "TWO ERRORS".

According to the decoding algorithm step 10;

$$S_1S_3 - (S_2)^2 = 2^4 - (2^8)^2 = 2^4-2^6= 2^7$$

We try to satisfy the equation;

$$2^7=(S_0 2^{2(9-l)}-S_1 2^{9-l})(S_1 2^{9-k}- S_0 2^{2(9-k)})$$

Substituting the values of S_0 and S_1, and applying the trial and error method, the only pair of k and l would be calculated as;

$$k = 3 \ , \ l = 4$$

So, decide the error positions as position numbers 3 and 4.

According to step 12 of the algorithm, pick the first possible pair of the e and d using Table V and then try to satisfy the equation described in step 13;

$$S_1 = e2^{9-k} + d2^{9-l}$$
$$2^3 = 1*2^{9-3} + 1*2^{9-4}.$$
$$2^3 = 2^6 + 2^5 = 8$$
$$2^3 = 2^3$$

So, it would take only one iteration to decide the errors, deciding e = d = 1. Set the error vector E as described in step 15 of the decoding algorithm;

E = 00011000000.

If e = d = 1 did not satisfy the equation, the next pair e = 3, d = A would be tried, etc.

Applying step 16 of the algorithm;

R = 0 0 0 1 1 0 1 -3 1 2 -1

\- E = 0 0 0 1 1 0 0 0 0 0 0

C = 0 0 0 0 0 0 1 -3 1 2 -1

Check and verify the syndromes and output the verified codeword as C = 0000001-312-1. Again the information is given as 0000001.

If more than 2 errors were made, then the above procedure would fail and a "decoding failure" would result. At this point a request for a retransmission would be initiated by the receiver.

VII. POSSIBLE INTERFACE FOR DATABASE APPLICATIONS

A. BACKGROUND

The size, power and number of database management information systems available and in use has grown dramatically in recent years. Business, industry and government seem to have been swept into the automation of data collections with a fervor akin to the automation of accounting systems during the late 1960's. As might be expected, people are once again discovering that the old adage of "garbage in, garbage out" continues to hold true. Although many of the errors in databases could probably be caught and corrected by appropriate error checking and correcting procedures, the cost for humans to perform such checking would be very high and the work very tedious. It makes sense to automate data storage and management, and remove the painstaking tasks for error checking and correcting from the human operator.

Virtually all DBMS's (Database Management System) incorporate in them some form of error checking facilities. [Ref. 7]. Typical checks are for proper data format (integer, real, alphabetic, etc.), proper numeric sign, the correct number of data items and the presence or absence of data in certain fields. While these checks are important and help to prevent some errors, the number of situations in which major errors in databases have been found clearly indicates that these checks are not sufficient. The purpose of the error correction mechanism we described in the previous chapters is to provide a means of error checking which far exceeds the power of typical DBMS type of error checking. The error correction mechanism, we introduced, can be used both to perform error correction for pre-existing databases, which we term the checking of of a 'static' database, and to check proposed database updates before they are

passed on to the DBMS, which we term the checking of a 'dynamic' database.

B. DATABASE ERRORS

Database errors can be divided into four types: security, consistency, reliability and integrity. Security encompasses the control of all unauthorized access to the database. Both physical and logical means of access control are usually required. Consistency deals with the problems of errors which are introduced in the process of sharing databases. These can be due to either multiple users sharing a single database or multiple users sharing more than one copy of a database. A database can be inconsistent when multiple updates are processed out of sequence or the database changes during the course of a user providing an update. Under this definition, consistency checking involves only the specific data. Such checking would not invoke any of the "implied meanings" of the data, i.e the information in the semantic description of the database. Thus, a requirement such as an update of one item necessitating a corresponding update of another item is not a consistency problem [Ref. 8]

Reliability refers to the problems of assuring that both the hardware and software components of the data management system perform as they were intended all of the time. Integrity errors include all types of errors which can be introduced due to active use of the database system. These may result from mundane sources such as typing or spelling errors, transmission errors which cause the data to be garbled or transformed between the original source and the database system, or user misunderstandings of the nature or content of the database.

A computer has no built-in criteria which it can use to determine whether or not a given piece of data is correct in a given context. Thus, if a computer is to be used to detect and correct integrity errors, the computer must be

provided with such criteria. The error correction mechanism we described addresses only the detection and correction of primarily the last of these error types, integrity errors, although some consistency errors can be detected.

C. QUERY INTERPRETER

Before starting to explain the role of the query interpreter which is the existing interface between the users and DBMS, it is necessary to identify the types of queries which could be given to the Supply System database.

Depending upon the NATO countries in which this kind of interface exists, the number of user queries which can be given to the system might vary. In general, there is a range of twenty to thirty queries which can be found in a typical Supply System including queries for search, update, delete etc. However, one common part of all these queries is the NSN (National or Nato Stock Number). In other words, no matter what the query is, there should be a stock number in it, since the stock number is the only key of the database. Thus, it would be a pretty good idea to check and then verify the NSN before it is passed to DBMS for processing according to the given query. This kind of checking can be obtained via a detection and correction routine which would be added to the query interpreter as a component.

As shown in Figure 7.1, the checking component of the query interpreter acts as a filter between users of the DBMS and DBMS itself. This checking component is a 'passive' filter in the sense that it is not visible to the DBMS users until a potential error is detected and corrected. However, whenever an error is detected and corrected, the system informs the user about the correction has been made, or if the number of errors are more than two it declares a decoding failure and requests a retransmission or another query.

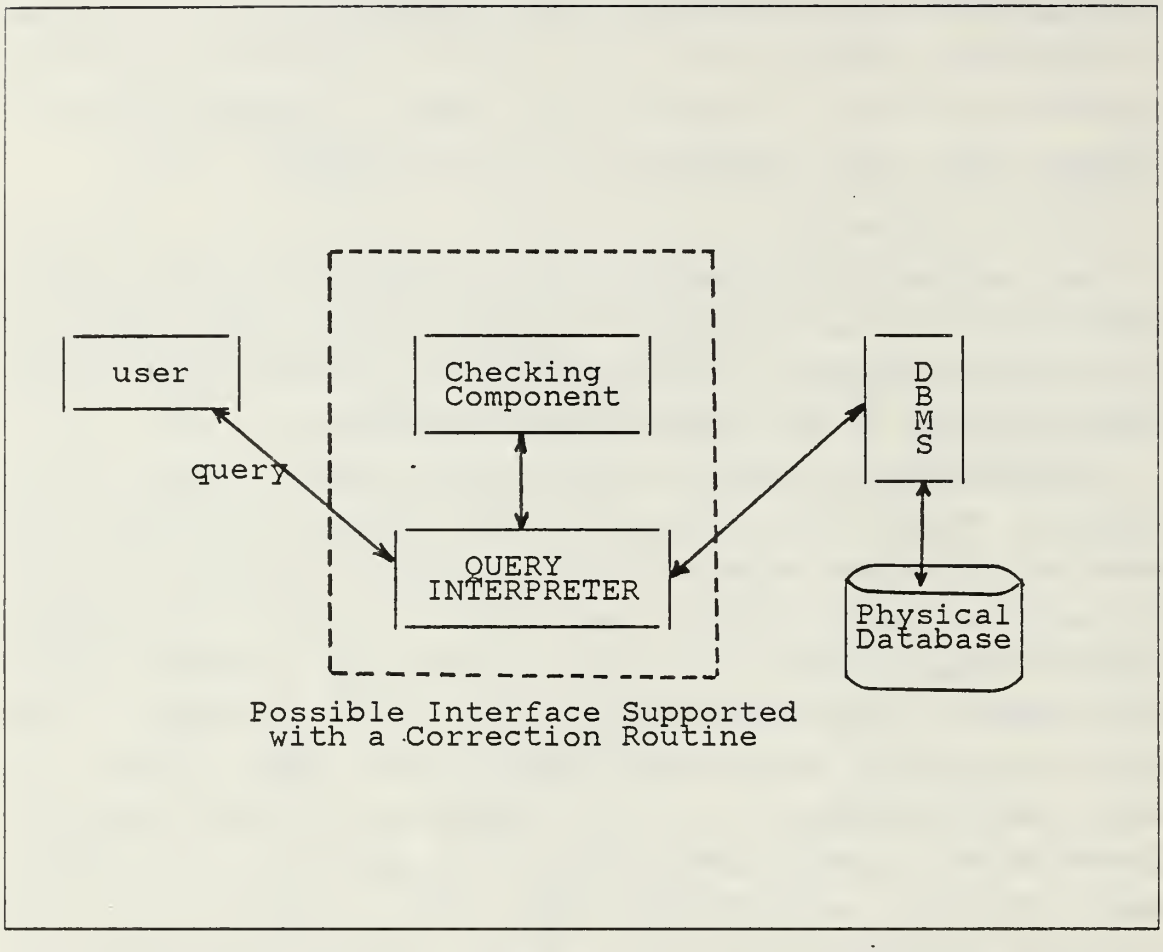


Figure 7.1 Possible interface between user and DBMS

Generally, writing detection routines is not difficult. If the structure is sufficiently well understood for update and access routines to be written, then the detection routine can likely also be written with about the same effort. The implementation of a correction routine is more difficult than implementation of a detection routine. But, for this particular application, it becomes relatively easy using the decoding algorithm we presented in Chapter VI.

Recalling the construction of a NSN from Chapter I, there are thirteen digits in a NSN. After the encoding process, there are added four more digits as check digits which makes the length of the codeword seventeen digits.

When this encoded codeword is received with or without error by the query interpreter, it is the correction routine's responsibility to check and verify it using the check digits and the decoding algorithm described in Chapter VI. In this way, terminal operator's mistakes as well as the other type of mistakes discussed as integrity errors can be detected and corrected. After verifying the received codeword (encoded NSN), the check digits are removed and the output of the correction routine gives the original thirteen digit NSN. Then this NSN and the interpreted query are passed to the DBMS for processing. Using this kind of interface and error checking and correcting mechanism together does not require any change in the construction of the physical database, and it provides a more efficient system in terms of reliability, integrity and time.

VIII. CONCLUSION

In this thesis we have taken a modular approach to the implementation of Reed-Solomon code in order to provide an error correction mechanism for the existing National or NATO Supply systems. By initially discussing algebraic coding theory and finite field theory, we have shown that they play an integral part in the overall implementation. The implementation theory is represented first because of its necessity to understand the implementation more easily. It is then followed by the design of the encoding and decoding algorithms which provide two error correction for the National or NATO Stock Number (NSN).

After defining the approach and the associated algorithms, it is then followed by a possible database interface. Thus the user of a database system can be supported by this kind of interface and database system itself becomes more reliable and efficient. The most common problems and general types of errors we have presented in the previous chapter showed that, when human operators get involved with the operation of these kind of systems, numerous types of errors should be expected. Encoding and decoding algorithms we presented in this thesis are developed based on the facts that an algorithm should be satisfied in order to detect and correct possible types of human operator errors without costing more in terms of money and personal effort. Because we can correct up to two errors, the reliability and integrity have improved.

It is hoped that, with this thesis as a guide, some interested supply officers or other officials will make the necessary changes in the Supply System Database in order for increasing its reliability and efficiency.

LIST OF REFERENCES

1. Peterson, W., Error Correcting Codes, M.I.T Press, Massachusetts, 1965.

2. Management Data List, Defense Logistic Services Center, Battle Creek, Michigan, 1985.

3. Balakrishnan, A.V., Communication Theory, McGraw-Hill Inc., New York, 1968.

4. McElice, R., Encyclopedia of Mathematics and Its Applications, Vol. 3, Addison-Wesley Publications, Massachusetts, 1977.

5. McKenzie, S.S., A Systolic Array Implementation of a RS Encoder and Decoder, Master's Thesis, Naval Postgraduate School, Monterey, California, 1985.

6. Hamming, R.W., Coding and Information Theory, Prentice-Hall Inc., New Jersey, 1980.

7. Ullmann, J.D., Principles of Database Systems, Computer Science Press Inc., Maryland, 1983.

8. Kroenke, D., Database Processing, Science Research Associates Inc., Chicago., 1983.

BIBLIOGRAPHY

Berlekamp, E.R., The Development of Coding Theory, IEEE Press, New York, 1974.

Pless, V., Introduction to the Theory of Error Correcting Codes Wiley-Inter Science Inc., New York, 1982.

Supply Department of NPS, Customer Service Manual, Naval Post Graduate School, Monterey, California, 1983.

Hartnett, W.E., Foundations of Coding Theory , D.Reidel Publishing Inc., Boston, 1975.

Berlekamp, E.R., Technology of Error Correcting Codes, Proc. IEEE, Vol. 68, May 1980, pp. 567-593.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5000	2
3.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93943-5000	1
4.	Professor H. Fredricksen, Code 53Fs Department of Mathematics Naval Postgraduate School Monterey, California 93943-5000	4
5.	Professor Thomas Wu, Code 53Wq Department of Computer Science Naval Postgraduate School Monterey, California 93943-5000	1
6.	Deniz Kuvvetleri Komutanligi Kutuphanesi Bakanliklar-Ankara Turkey	1
7.	Deniz Egitim Komutanligi Kutuphanesi Karamursel Kocaeli Turkey	1
8.	Deniz Harb Okulu Komutanligi Kutuphanesi Tuzla Istanbul Turkey	1
9.	Mehmet Yenen Bati Garnizon Lojmanlari Batiray Apt. B Blok No.6 Yuzbasilar-Golcuk Kocaeli Turkey	1
10.	Husnu Emekli Inonu Cad.Subay Loj. No. 441/8 Hatay-Izmir Turkey	1

219431

Thesis
Y356
c.1

Yenen
An approach to the concept of error recovery in the National Stock Number System.